תרגיל בית תיכנותי להגשה

עד 04.02.2025 בשעה 23:59
בהצלחה!

תרגיל זה מנוסח בלשון זכר מטעמי נוחות בלבד והוא מיועד לכל המגדרים.
מתרגל אחראי על התרגיל: יהונתן פסחובסקי

הוראות:

- יש להגיש קובץ zip יחיד כאשר השם של הקובץ הוא תעודות זהות של חברי הקבוצה המופרדים על ידי קו תחתון למשל אם שני שותפים עם ID1, ID2 אז השם של הקובץ יהיה ID1_ID2.zip.
- ההגשה תתבצע רק ע"י אחד מבני הזוג למקום הייעודי באתר הקורס במודל.
- עליכם לוודא לפני ההגשה במודל כי הקוד שלכם מתקמפל ורץ בשרת **Microsoft Azure** שהוקצה לכם (הוראות מצורפות בקובץ נפרד).
- זוג שהתרגיל שלו לא יתקמפל בשרת שהוקצה או יעוף בזמן ריצה ציונו בתרגיל יהיה 0.
- יש לכתוב קוד קריא ומסודר עם שמות משמעותיים למשתנים, למתודות ולמחלקות.
- יש להקפיד למלא את כל דרישות התרגיל (שימוש בייצוג נכון, סיבוכיות זמן וכו'). אי עמידה בדרישות התרגיל תגרור ציון 0.

תיאור כללי

בעולם המהיר של מסחר במניות, ניהול ומעקב אחרי מחירי המניות באופן יעיל הוא קריטי. המשימה שלכם היא לעצב מערכת המאפשרת עדכונים בזמן אמת ושאילתות לגבי מחירי מניות. המערכת צריכה להתמודד עם כמות גדולה של נתונים תוך שמירה על ביצועים אופטימליים עבור מגוון פעולות.

פרטים

מניה היא בעלות של חלק מחברה מסוימת, מזוהה באופן ייחודי על ידי stockId שלה, שהוא מזהה המנייה.

לכל מניה יש מחיר, שיכול להשתנות עם הזמן, והמערכת עוקבת אחרי השינויים הללו באמצעות חותמות זמן, שמייצגות רגעים ספציפיים שבהם מתבצעים עדכוני מחירים.

במחירי מניות יש שני אירועים:

- אירוע איתחול מחיר: במהלך אירוע זה מחיר המניה מאותחל, וזה מחירה ההתחלתי.
- אירוע עידכון מחיר: במהלך אירוע עידכון מחיר, מחיר המניה משתנה (עולה או יורד).

אירוע עידכון מחיר עלול להיפסל בדיעבד, ועל המערכת להתעלם מאירוע זה החל מרגע פסילתו. עקב כך, כל אירוע עידכון מחיר יקרא רלוונטי, עד שהמערכת תקבל עידכון שהוא נפסל (יתכן גם שלעולם לא יפסל) ואז הוא יקרא אירוע עידכון מחיר לא רלוונטי. אירוע איתחול מחיר לא יכול להיפסל, ותמיד יהיה רלוונטי.

מחיר המנייה הנוכחי הוא המחיר של המניה ברגע האיתחול פלוס סכום כל אירועי העידכון הרלוונטים.

הערות חשובות לתרגיל:

- לכל אורך התרגיל נסמן בN את מספר המניות הכולל וב$M_{stockId}$ את מספר אירועי העידכון הרלוונטיים שיש למניה שהמזהה שלה הוא stockId .
- אלא אם נאמר אחרת לא ניתן להניח שהקלט תקין. במקרה והקלט לא תקין (לדוגמא מחיקת מנייה שלא קיימת,הוספת מנייה קיימת וכולי) יש לזרוק חריגה מסוג **IllegalArgumentException**.

עליכם ליצור קובץ בשם StocksManger.java שבו תממשו מחלקה פומבית בשם StocksManger שבה יהיה מבנה הנתונים המאפשר את ניהול המערכת לניתור המניות.

<u>על מבנה הנתונים שלכם לתמוך בפעולות הבאות:</u>

1. `void initStocks()`

   מאתחל את מערכת המעקב אחרי המניות.

   סיבוכיות זמן: $O(1)$

2. `void addStock(String stockId, long timestamp, Float price)`

   אירוע איתחול מחיר: מוסיף למערכת מניה חדשה שהמזהה שלה הוא stockId , המחיר ההתחלתי שלה הוא price וחותמת הזמן של האירוע הוא timestamp .

   סיבוכיות זמן:$O(log(N))$

3. `void removeStock(String stockId)`

   מסיר מהמערכת את המניה שהמזהה שלה הוא stockId ואת כל המידע ההיסטורי שלה.

   סיבוכיות זמן: $O(log(N))$

4. `void updateStock(String stockId, long timestamp, Float priceDifference)`

אירוע עידכון מחיר: מזין למערכת אירוע עידכון מחיר חדש שבו מניה, שהמזהה שלה הוא stockId , משנה את המחיר שלה כך שהערך priceDifference (שיכול להיות חיובי או שלילי) מתווסף למחיר המניה הנוכחי, וחותמת הזמן של האירוע היא timestamp. ניתן להניח כי חתימת הזמן מאוחרת יותר מכל החתימות הקודמות שניתנו למניה.

סיבוכיות זמן:$O(log(N) + log(M_{stockId}))$

5. `Float getStockPrice(String stockId)`

מחזיר את המחיר הנוכחי של מניה שהמזהה שלה הוא stockId .

סיבוכיות זמן: $O(log(N))$

6. `void removeStockTimestamp(String stockId, long timestamp)`

מוחק מהמערכת אירוע עידכון מחיר פסול של המניה שהמזהה שלה הוא stockId ושחותמת הזמן של האירוע היא timestamp.

סיבוכיות זמן:$O(log(N) + log(M_{stock}))$

7. `int getAmountStocksInPriceRange(Float price1, Float price2)`

סופר את כמות המניות שמחירן הנוכחי p מקיים $price1 \leq p \leq price2$.

סיבוכיות זמן:$O(log(N))$

8. `String[] getStocksInPriceRange(Float price1, Float price2)`

מחזיר רשימה של מזהי מניות שמחירן הנוכחי p מקיים $price1 \leq p \leq price2$, ממוינת לפי מחירן.

סיבוכיות זמן: $O(log(N) + K)$ - K הוא מספר המניות בטווח.

<u>הבהרות לגבי התרגיל</u>

- שבירת שיוויון נעשת על פי ה stockId. כלומר, אם לשתי מניות יש אותו מחיר, ושתיהן מוחזרות מהפונקציה getStocksInPriceRange, אז הסדר שלהן ברשימה יהיה על פי הסדר האלפבתי של stockId.
- המחיר הראשוני של מנייה חייב להיות חיובי ממש (גדול מ0).
- עידכון המחירים של המנייה חייבים להיות שונים מ0. כלומר $priceDifference \neq 0$.
- המערך אשר מוחזר מהמתודה getStocksInPriceRange צריך להיות בדיוק בגודל של מספר המניות אשר יש בטווח זה. כלומר אין להחזיר מערך אשר גם מכיל ערכים אחרים (כגון null וכולי).

<u>הסבר על הקבצים שקיבלתם</u>

- StockManger.java - זה הקובץ שבו אתם כותבים את הקוד שלכם.
- Main.java - הקובץ שבו תרוץ בדיקת התרגיל.

<u>מבנה ההגשה</u>

- יש להגיש קובץ zip אשר מכיל את הקובץ StockManger.java וכל קובץ אחר שמימשתם כדי לפתור את התרגיל.
- קוד אשר ישתמש בקבצים אחרים שלא יהיו בzip יקבל 0.

<u>אילוצים</u>

- הקוד אינו יכול להכיל import לשום מחלקה שלא מימשתם.
- אין להשתמש במחלקה System (אין לרשום בקוד שאתם יוצרים .System).